INSIDE VISUAL dBASE™

July 1996 • Vol. 3 No. 7
US $8.50

Tips & techniques for Visual dBASE and dBASE for Windows

Visual dBASE Specific

dWorld Wide Web: A first look

by Keith Chuvala

Oh, what tangled web we weave, when first we practice to retrieve. Or something like that. Bringing database applications to the Internet's World Wide Web is a hot topic these days, and for good reason. Everybody loves (or thinks they should love) the Web. In this article, we'll show you how to make Web pages communicate with Visual dBASE applications.

Internet 101: Gaining access

For those of you who haven't yet experienced the Internet and the World Wide Web, this love affair may be difficult to understand. If you're one of the five or six people who still aren't on the Internet in some way, I encourage you to find a way to get connected. You can obtain an Internet account through an established online service like CompuServe, America Online, or Prodigy for a flat monthly rate from under $10 on up, depending on the class of service you desire.

For bona fide Net junkies or cheapskates (I happen to be both), a local Internet Service Provider (ISP) typically offers either metered or flat-rate service. Local services often provide much better performance than the big services, with the added benefit of local access phone numbers—a real advantage for folks in smaller towns. You won't get all the nifty features offered by the big online services, but you'll have a faster and sometimes less expensive connection to the Internet.

Many companies have already invested in—or are currently in the process of—buying or leasing high-speed circuits connected directly to the Internet. These folks have discovered that making databases available over the Internet is a very attractive way to give users access to corporate data. With the Web, users can get to data easily, no matter where they live, no matter what hardware they use, and no matter what kind of Internet connection they make.

Developing for the Web

Developers like the Web because its various components—including HTML (hypertext markup language, the language used to compose Web pages) and HTTP (hypertext transfer protocol)—are elegant and easy to learn. With these tools, programming for the Web is relatively simple. Even unsophisticated users like the Web because, to get to the data they need, they must master only their Web browser software rather than a collection of strange, different-acting application packages.

So why isn't everyone happy yet? There are two reasons why marrying database applications to Web front-ends isn't a straightforward process. First, the programs that run the Web are usually in a different place than the programs that comprise a company's database systems.

IN THIS ISSUE

Most Web servers run on UNIX-based machines, while databases frequently reside on DOS- or Netware-based LANs, or even large mainframes.

Second, HTML is less than perfect at handling database access. Such matters are left to a somewhat cryptic mechanism called the Common Gateway Interface, or CGI. CGI programs are typically UNIX shell scripts, compiled C programs, or interpreted Perl scripts. While you can work wonders with databases using CGI tools, existing DOS- and Netware-based LANs and large mainframe database systems rarely use them. For a business or organization that uses Visual dBASE for its database management, we need to deal with both problems.

dBASE-powered Web sites

Borland International has promised that the upcoming 32-bit release of Visual dBASE will include Web tools, and I certainly look forward to those. In the meantime, though, I did some research using the current edition of dBASE, along with off-the-shelf tools from other vendors. My goal was to create a running Web site using Visual dBASE for all database activities. I gave myself a week to build it—it took less than a day!

Now, before you head for the nearest FTP site looking for software to download, let me offer an up-front reality check. First, I'm already very familiar with the Web and its tools, since I'm the Webmaster for the college I work for. I know UNIX, I know HTML, I know TCP/IP, and I know the people who know the things I don't yet know. If you're new to the Web itself, to HTML, or to Visual dBASE, you'll have to invest time to learn all of the above before you can create useful Web-based database applications.

Table A: *Commonly used HTML tags*

HTML Tag	What it does
`<B>...</B>`	Begin/end boldface
`<BODY>...</BODY>`	Begin/end body of document
`<BR>`	Line break
`<FORM>...</FORM>`	Begin/end HTML input form
`<H1>...</H1>`	Heading style 1
`<H2>...</H2>`	Heading style 2
`<H3>...</H3>`	Heading style 3
`<H4>...</H4>`	Heading style 4
`<H5>...</H5>`	Heading style 5
`<H6>...</H6>`	Heading style 6
`<HEAD>...</HEAD>`	Begin/end page heading
`<HR>`	Horizontal rule (centered by default)
`<I>...</I>`	Begin/end italics
`<IMG SRC="image.gif">`	Inline graphic image
`<HTML>...</HTML>`	Begin/end HTML document
`<LI>`	Line item for <OL> and <UL> lists
`<OL>...</OL>`	Begin/end ordered (numbered) list
`<P>`	Paragraph break
`<PRE>...</PRE>`	Begin/end preformatted text
`<TABLE>...</TABLE>`	Begin/end table
`<TD>...</TD>`	Begin/end table data "cell"
`<TITLE>...</TITLE>`	Begin/end document title
`<TR>...</TR>`	Begin/end table row (</TR> is optional)
`<UL>...</UL>`	Begin/end unordered (bulleted) list

What's in a Web page?

World Wide Web pages are plain-text files containing Hypertext Markup Language (HTML) tags. These tags, which consist of instructions set in angle brackets (< and >), dictate the appearance and behavior of the Web page. **Table A** lists a number of commonly used HTML tags that you'll find in typical Web pages.

One site that contains numerous links to helpful resources for folks who are new to this stuff is located at

```
http://home.netscape.com/home/how-to-
    create-web-services.html
```

Netscape corporation maintains this site , and it's very up-to-date. If you're brand-new to Web page authoring and you don't have a good book on HTML, *A Beginner's Guide to HTML* is a must-read. You can find the *Beginner's Guide* online, of course, at

```
http://www.ncsa.uiuc.edu/General/Internet/
    WWW/HTMLPrimer.html
```

(These sites were active at the time of this journal's publication.)

Web server software

I run Windows for Workgroups 3.11 on my home development machine, which proved a bit problematic for building my Web site. It seems that almost all of the new development in Web server software is aimed at the Windows 95 and Windows NT platforms. Fortunately, there are still some offerings for 16-bit Windows. The one I chose for this first dBASE project is W4-Server from Antelope Software (*http://130.89.224.16/*). Antelope also produces Win95 and NT versions of W4-Server, and it advertises each of those offerings as having more features and—as you would expect from 32-bit products—improved performance. However, the 16-bit version I downloaded runs quite well with trusty old 16-bit Windows.

W4-Server installs itself cleanly and is easy to get up and running. The shareware registration fee is very reasonable at $25 per copy. It's not the only software out there, but it's very well suited to exploring the potential of integrating Visual dBASE with the Web.

I created a top-level Web page named INDEX.HTM (see Listing E on page 7) using simple HTML tags, started a SLIP connection to my Internet provider, and ran W4-Server through some simple tests. Figures A and B show the resulting page as viewed in Netscape. The W4-Server program is running minimized behind the scenes. I'm running both the client (Netscape) and server (W4-Server) processes, so I can test any changes to the code immediately. Doing so also assures me that I'm seeing exactly what any end-users will see, as long as they run Netscape or a similar graphical Web browser.

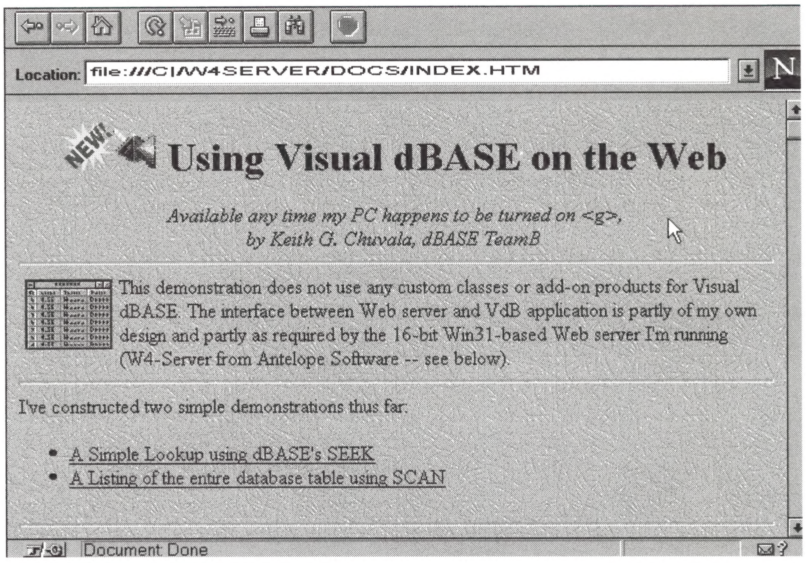
The Web form

Listing A, on the following page, contains the code for SWQUERY.HTM, the first Web page I wrote for this project. It uses an HTML form to get data from the user. Don't confuse HTML forms with dBASE forms! An *HTML form* is a region of a Web page populated with input and editing controls.

Each form will feature at least one pushbutton or "hot spot" (for text-only browsers). Clicking the button or selecting the hot spot causes a specified program to run, and this is how we make our Web pages talk to Visual dBASE applications. In SWQUERY.HTM, the line that specifies the dBASE application is

```
<FORM ACTION="/swquery.wtd" METHOD=POST>
```

Figure A


Location: file:///C|/W4SERVER/DOCS/INDEX.HTM

Using Visual dBASE on the Web

Available any time my PC happens to be turned on <g>, by Keith G. Chuvala, dBASE TeamB

This demonstration does not use any custom classes or add-on products for Visual dBASE. The interface between Web server and VdB application is partly of my own design and partly as required by the 16-bit Win31-based Web server I'm running (W4-Server from Antelope Software -- see below).

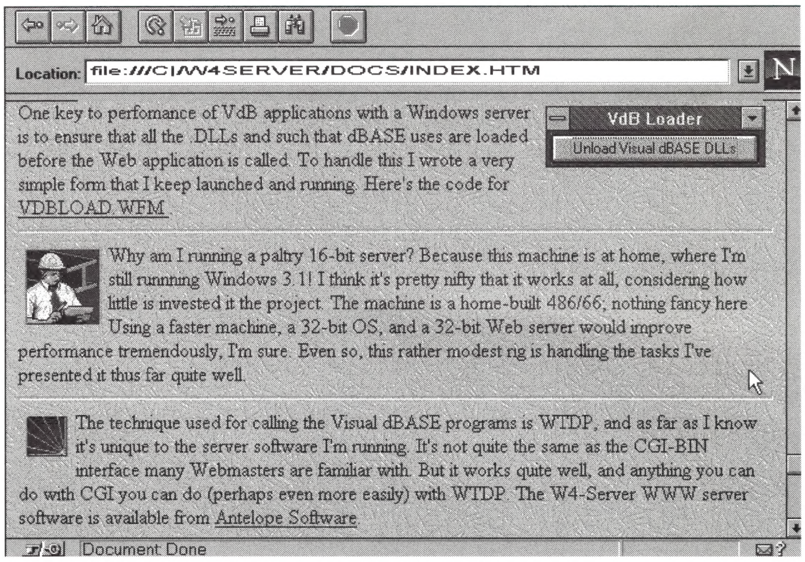
I've constructed two simple demonstrations thus far:

- A Simple Lookup using dBASE's SEEK
- A Listing of the entire database table using SCAN

Document: Done

The top half of our top-level Web page, INDEX.HTM, looks like this when viewed from Netscape.

Figure B


Location: file:///C|/W4SERVER/DOCS/INDEX.HTM

One key to perfomance of VdB applications with a Windows server is to ensure that all the .DLLs and such that dBASE uses are loaded before the Web application is called. To handle this I wrote a very simple form that I keep launched and running. Here's the code for VDBLOAD.WFM.

Why am I running a paltry 16-bit server? Because this machine is at home, where I'm still runnning Windows 3.1! I think it's pretty nifty that it works at all, considering how little is invested it the project. The machine is a home-built 486/66; nothing fancy here. Using a faster machine, a 32-bit OS, and a 32-bit Web server would improve performance tremendously, I'm sure. Even so, this rather modest rig is handling the tasks I've presented it thus far quite well.

The technique used for calling the Visual dBASE programs is WTDP, and as far as I know it's unique to the server software I'm running. It's not quite the same as the CGI-BIN interface many Webmasters are familiar with. But it works quite well, and anything you can do with CGI you can do (perhaps even more easily) with WTDP. The W4-Server WWW server software is available from Antelope Software.

Document: Done

This is the bottom half of our top-level Web page.

The ACTION clause of the FORM tag specifies a file that W4-Server uses to tie the page and processing program together. If your server software supports only CGI for calling programs, specify the particular application's filename instead of the WTD file. SWQUERY.WTD appears in Listing B on the next page.

You should name each input field on an HTML form. The name is technically optional, but for any form with more than

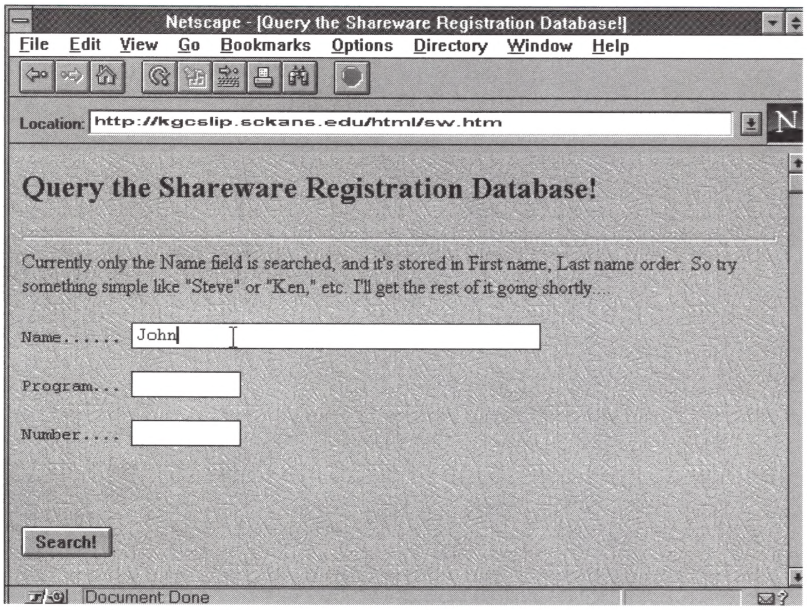
Listing A: *SWQUERY.HTM*

```
<html>
<head>
<title>Query My Bogus Shareware Registration Database!</title>
</head>
<body background="relief.gif">
<h2>Query My Bogus Shareware Registration Database!</h2>
<hr>

Currently only the Name field is searched, and it's stored in
First name, Last name order.  So try something simple like "Steve"
or "Ken," etc.  I'll get the rest of it going shortly....<P>

<FORM ACTION="/swquery.wtd" METHOD=POST>
<pre>
Name...... <input type="text" name="username" size=40,2 maxlength=40><br>
Program... <input type="text" name="program"  size=10,2 maxlength=10><br>
Number.... <input type="text" name="number"   size=10,2 maxlength=10><br>
</pre>
<p>
<input type="submit" value="Search!">.
<p>
</CENTER>
</body>
</html>
```

Listing B: *SWQUERY.WTD*

```
[WTDP]
ExecFile    swquery.exe
OutputFile  \html\results.htm
HTMLFile    \html\results.htm
[END]
```

Figure C


Netscape - [Query the Shareware Registration Database!]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://kgcslip.sckans.edu/html/sw.htm
Search!
Document: Done

Query the Shareware Registration Database!

Currently only the Name field is searched, and it's stored in First name, Last name order. So try something simple like "Steve" or "Ken," etc. I'll get the rest of it going shortly...

Name...... John

Program...

Number....

The input form looks like this to a Netscape user.

one input field, we'll need a name to distinguish between fields. This name has no direct relationship to the display the user sees; it's simply there for your convenience in processing the form. Figure C shows the Web form as seen by a Netscape user.

The dBASE application

SWQUERY.PRG, shown in Listing C on page 6, is the Visual dBASE application that does most of the work. The GetParm( ) function extracts a parameter from the data file that W4-Server creates when the user clicks the pushbutton. For this application, we grab the value of the USERNAME input field that's contained in the Web form SWQUERY.HTM.

I won't delve into all the details of this program. Much of it is easy to follow, though the output side deserves some attention. The program writes all output to a text file named RESULTS.HTM, which I specified earlier in the WTD file. This is the file that W4-Server will load after our dBASE application finishes running. I like to use dBASE's low-level I/O functions for this kind of work, and in this application a single fputs( ) writes the results of the search to the output file.

When the search is successful, the program writes data from the located record along with appropriate HTML code to produce the resulting display, illustrated in Figure D. If the search was unsuccessful, the program writes a helpful error message in HTML format, as shown in Figure E. Notice that no matter what the result, the Web server software will display RESULTS.HTM, as specified in the SWQUERY.WTD file we discussed earlier. All output is produced dynamically by the dBASE program!

Maximizing performance

Visual dBASE-generated EXE files are fairly small. The runtime support they require, however, is quite large. This demand makes for relatively long load times for a typical dBASE application. Each time a user clicks the Search button on our query form, all that code has to be loaded from disk, making for pretty disappointing performance.

To avoid this problem, I created a small form called VDBLOAD.WFM, shown in

Listing D on page 7, and compiled it to an EXE. I launched this program along with the Web server software, ensuring that the DLL and other resources my Visual dBASE applications required would already be loaded in memory when the Web page called the EXE program file. Doing so noticeably improved overall performance.

Any dBASE applications designed to work with the Web should be as small and fast as possible. There are enough bottlenecks on the Internet already—no need for our database applications to add to users' frustration!

What's next for dWeb?

Our example form is of limited use because it's hard-coded for a particular table in a particular situation. A far more useful tool would allow us to specify the table or query, and so on, in the HTML form or other calling program. Borland promises to deliver just that kind of tool with Web Tools for Visual dBASE, a set of custom classes that should make creating applications for the World Wide Web easier than it is now. Web Tools will require far less duplication of effort to create hard-coded forms and applications like those discussed in this article.

When Web Tools becomes available, we'll examine it here in *Inside Visual dBASE*. In the meantime, I'd love to hear from you on this topic. Are you using the Web for database work now? If not, will you need to do so in the near future? What kind of information would you like to read in these pages to make your job easier or to raise your level of understanding of how to integrate Visual dBASE with the Web? See "Subscriber Survey" on page 16 for more information about sharing your input regarding this journal.

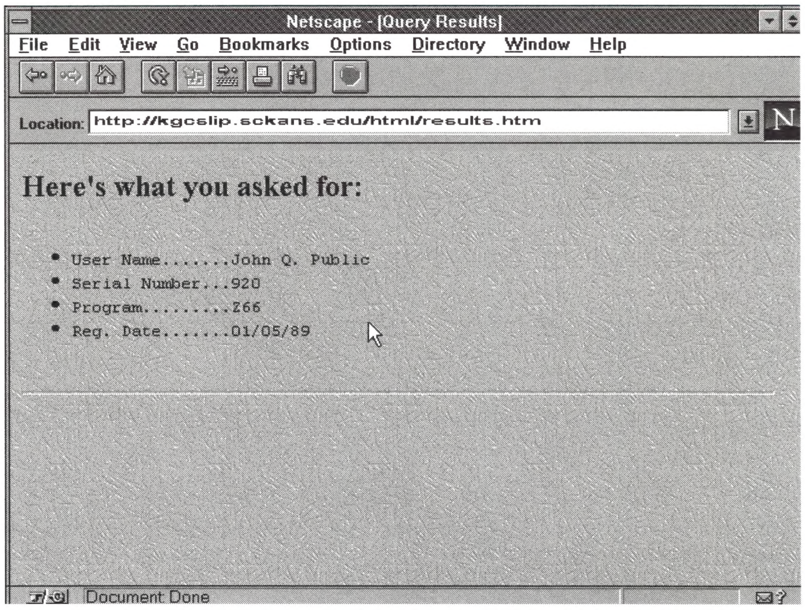
Try it yourself!

The dBASE and HTML code we present in this article, and more like it, is available on my home Web server, which will soon be moving to a new location. The best way to find it is to go to my permanent home page at

```
http://www.sckans.edu/~kgc.
```

You'll find a link to the pages presented here, as well as other nifty Visual dBASE-related links. Stop by and check it out! ❖

Figure D

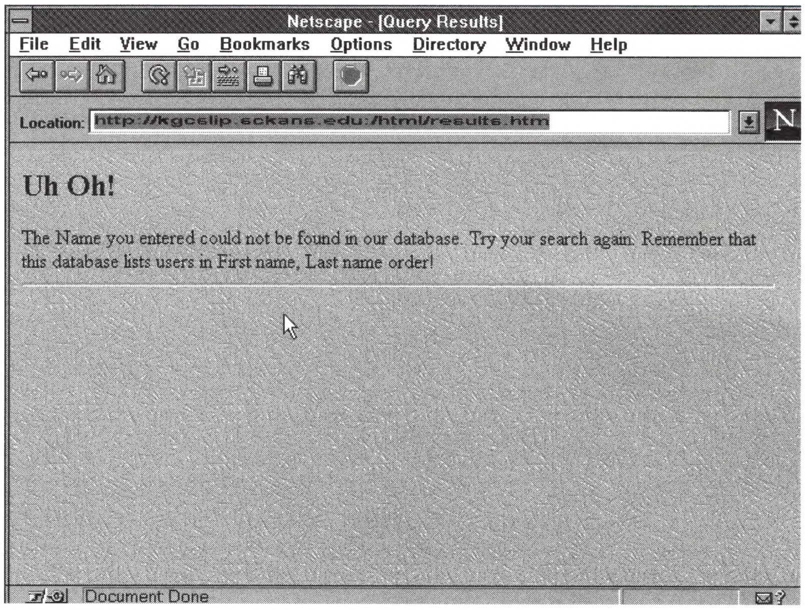

Netscape - [Query Results]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://kgcslip.sckans.edu/html/results.htm
Document Done

Here's what you asked for:

```
• User Name.......John Q. Public
• Serial Number...920
• Program.........Z66
• Reg. Date.......01/05/89
```

A successful SEEK in dBASE produces a nicely formatted Results screen.

Figure E


Netscape - [Query Results]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://kgcslip.sckans.edu:/html/results.htm
Document Done

Uh Oh!

The Name you entered could not be found in our database. Try your search again. Remember that this database lists users in First name, Last name order!

An unsuccessful search results in this HTML error message.

Listing C: *SWQUERY.PRG*

```
* SWQUERY.PRG: Gathers Web data from Web form and writes
*              output to target HTML file
parameter coutfile
set talk off
local i
cParms = ""
cParm = ""
set safety off
nH = fopen("\windows\w4-serve.tmp")
cs = upper(fgets(nH))
fclose(nH)

cUser = getparm("USERNAME",cs)
CrLf = chr(13)+chr(10)
if type("coutfile") = "C"
   nH = fcreate("c:\w4server\docs\html\"+coutfile,"w")
else
   nH = fcreate("c:\w4server\docs\html\results.htm","w")
endif

use \ivdb\vdbweb\sw noupdate order name
set exact off
seek upper(ltrim(trim(cUser)))
if .not. eof()
   fputs(nH,"<html><title>Query Results</title><body
background=relief.gif>" + crlf + ;
          "<h2>Here's what you asked for:</h2>"      + crlf + ;
          "<pre><ul>"                                + crlf + ;
          "<li>User Name.......";+sw->name           + crlf + ;
          "<li>Serial Number..."+sw->serialno        + crlf + ;
          "<li>Program........."+sw->program         + crlf + ;
          "<li>Reg. Date......."+dtoc(sw->regdate)   + crlf + ;
          "</ul></pre><hr></body></html>")
else
      fputs(nH,"<html><title>Query  Results</title><body
background=relief.gif>" + crlf + ;
          "<h2>Uh Oh!</h2>"      + crlf + ;
          "The Name you entered could not be found in our " + ;
          "database.  Try your search again.  Remember that " + ;
          "this database lists users in First name, Last name " + ;
          "order!" + crlf + ;
          "<hr></body></html>")

endif
use
fclose(nH)
return

procedure getparm
parameter cWhat,cWhere
   local ctemp,p
   cTemp  = ltrim(trim(upper(cWhat)))
   cWhere = upper(cWhere)
   p      = at(cTemp,cWhere)
   cTemp  = substr(cWhere,p+len(cWhat)+1)
   cTemp  = left(cTemp,at("&",cTemp)-1)
return cTemp
```

Quick Tip

Knowing where home is

Visual dBASE contains many system memory variables that track various options within the program. One such memory variable tracks where your end user initially loaded Visual dBASE. The *_dbwinhome* memory variable contains the primary directory name where Visual dBASE resides.

However, this variable doesn't specify the name of the subdirectory where the Visual dBASE executable file resides. To locate that file, you must call the HOME( ) function, which returns the complete directory and subdirectory location of the Visual dBASE executable file. To examine the difference between these two methods from the Command window, enter the commands

```
? _dbwinhome
? HOME()
```

On our system, the memory variable contained the string *C:\VISUALDB\*, the name of the primary Visual dBASE directory. However, the HOME( ) function returned the string *C:\VISUALDB\BIN\*, the complete location of the Visual dBASE executable.

Listing D: *VDBLOAD.WFM*

```
** END HEADER * do not remove this line*
* Generated on 03/16/96
*
parameter bModal
local f
f = new VDBLOADFORM()
if (bModal)
   f.mdi = .F. && ensure not MDI
   f.ReadModal()
else
   f.Open()
endif
CLASS VDBLOADFORM OF FORM
   this.Top = 22
   this.ColorNormal = "RB"
   this.EscExit = .F.
   this.Text = "VdB Loader"
   this.Height = 1.6465
   this.Width = 29
   this.Maximize = .F.
   this.MDI = .F.
   this.Sizeable = .F.
   this.Left = 1

   DEFINE PUSHBUTTON PB_CLOSE OF THIS;
       PROPERTY;
         Top 0.1172,;
         Text "Unload Visual dBASE DLLs",;
         Height 1.4102,;
         Width 27.833,;
         FontBold .F.,;
         OnClick {; form.release()},;
         Group .T.,;
         Left 0.5

ENDCLASS
```

Downloading the 5.5A patch

The Visual dBASE 5.5A patch is out. You can find it on the Internet at Borland International's FTP archive

ftp://ftp.borland.com/pub/techinfo/techdocs/database/dbase/win/vdb/software/vdb55a/VDB55A.ZIP

In addition, you can download it from

ftp://ftp.sckans.edu/pub/dbase/vdb55a.zip

The patch is a Zip file—just unzip it in your Visual dBASE directory. This patch is free, and it fixes a number of known bugs in Visual dBASE 5.5.

Listing E: *INDEX.HTM*

```
<!* INDEX.HTM: top-level page for Visual dBASE/WWW project *>
<html>
<Title>Windows-Based Web Server with Visual dBASE!</title>
<body background=relief.gif>
<center>
<h1><IMG SRC="/pics/rnew.gif"><img src="vdbicon.gif">
Using Visual dBASE on the Web</h1>
<i>Available any time my PC happens to be turned on &ltg&gt,<br>
by Keith G. Chuvala, dBASE TeamB</i>
</center>
<hr>

<img src="browse.jpg" align=LEFT hspace=5 VSPACE=3>
This demonstration does not use any custom classes or add-on products
for Visual dBASE. The interface between Web server and VdB application
is partly of my own design and partly as required by the 16-bit
Win31-based Web server I'm running (W4-Server from Antelope Software *
see below).

<hr>

I've constructed two simple demonstrations thus far:

<uL>
<li><a href="http://kgcslip.sckans.edu/html/sw.htm">A Simple Lookup
    using dBASE's SEEK</a>
<li><a href="http://kgcslip.sckans.edu/html/swbr.htm">A Listing of the
    entire database table using SCAN</a>
</ul>

<hr>

<a href="./vdbload.htm"><img src="vdbload.jpg" align=right hspace=5
border=0 VSPACE=5></a> One key to performance of VdB applications with a
Windows server is to ensure that all the .DLLs and such that dBASE uses
are loaded before the Web application is called. To handle this I wrote
a very simple form that I keep launched and running. Here's the code
for <a href="./vdbload.htm"> VDBLOAD.WFM </a>.

<hr>

<img src="undercon.jpg" align=LEFT hspace=5 VSPACE=3> Why am I running a
paltry 16-bit server?  Because this machine is at home, where I'm still
running Windows 3.1! I think it's pretty nifty that it works at all,
considering how little is invested in the project. The machine is a
home-built 486/66; nothing fancy here. Using a faster machine, a 32-bit
OS, and a 32-bit Web server would improve performance tremendously, I'm
sure. Even so, this rather modest rig is handling the tasks I've
presented it thus far quite well.

<hr>

<A HREF="HTTP://130.89.232.242/"><img src="w4server.gif" align=left
hspace=5 vspace=3 border=0></a>

The technique used for calling the Visual dBASE programs is WTDP, and
as far as I know it's unique to the server software I'm running. It's
not quite the same as the CGI-BIN interface many Webmasters are familiar
with. But it works quite well, and anything you can do with CGI you can
do (perhaps even more easily) with WTDP. The W4-Server WWW server
software is available from <A HREF="HTTP://130.89.232.242/">Antelope
Software</a>.

<hr>

Return to my "real" home page at <a href="http://www.sckans.edu/~kgc">
Southwestern College.</a>

</html>
```

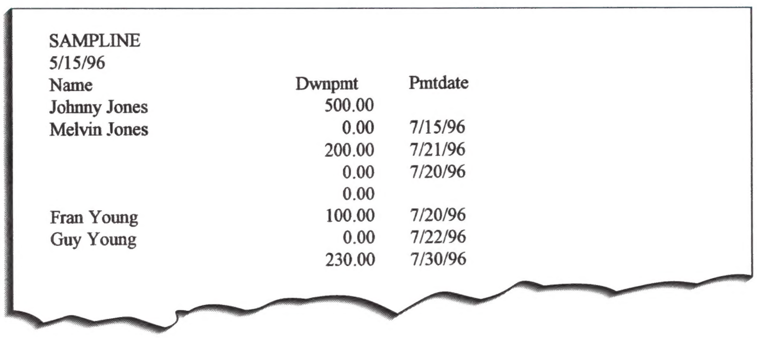
Report Tip

Printing underlines instead of spaces to make proofreading easier

If you create and maintain large-scale databases, you probably include some kind of quality assurance procedure in the data entry process. At the program-design level, you could build into your application some routines that prevent users from leaving required fields blank. However, in such applications, you have to balance your desire for complete data with the fact that, your end users don't always have all the information they need to enter complete, error-free records.

Suppose all of the data you manage doesn't come from your cozy application. You can acquire incomplete databases from outside sources or from data entry operators—working from the Command window—entering records on standalone systems. In those cases, it doesn't hurt to double-check the data before you start reporting on it.

Figure A

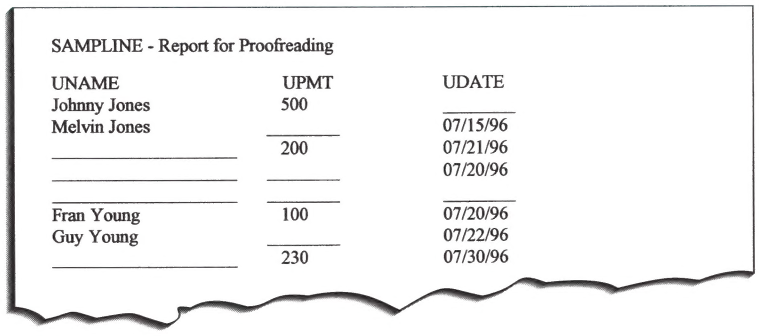


SAMPLINE
5/15/96

Name	Dwnpmt	Pmtdate
Johnny Jones	500.00	
Melvin Jones	0.00	7/15/96
	200.00	7/21/96
	0.00	7/20/96
	0.00	
Fran Young	100.00	7/20/96
Guy Young	0.00	7/22/96
	230.00	7/30/96

In a typical report, you can identify empty fields only by the blank space in a column.

Figure B



SAMPLINE - Report for Proofreading

UNAME	UPMT	UDATE
Johnny Jones	500	________
Melvin Jones	________	07/15/96
________________	200	07/21/96
________________	________	07/20/96
________________	________	________
Fran Young	100	07/20/96
Guy Young	________	07/22/96
________________	230	07/30/96

If you customize your report form, you can print underlines in place of blank fields to make proofreading easier.

Perhaps the best method for ensuring quality data entry is good, old-fashioned proofreading. After one person enters and prints a batch of records, your proofreader verifies accuracy by comparing the printouts to the source documents. That way, you can be sure the data you're storing is as complete and correct as possible.

However, suppose an end user gives you a printout that contains illegible or ambiguous changes? When you're posting changes to a table, the last thing you want to do is fret and worry over which handwritten note applies to which record.

Fortunately, Crystal Reports for Visual dBASE provides a number of ways to make your printed data easy to proofread and edit. For example, if you print underlines instead of empty fields in your report, your proofreader can quickly locate fields that need attention. To illustrate, compare the printed reports shown in Figures A and B.

We used Crystal Reports for dBASE's Expert to create the report shown in Figure A. In Figure B, we created a custom report so we could print underlines for empty fields. In this article, we'll show you how to create such a custom report.

Introducing the Formula Field

In brief, our technique is to use Crystal Reports for dBASE's Formula Field option to place Formula Fields—supercharged versions of dBASE for DOS's calculated fields—in the report instead of using regular table fields. It takes a little longer to enter Formula Fields, but the extra work is worth the trouble.

When you enter the custom expression for the Formula Field, you use IIF( ), the "immediate if function," to return a series of underlines instead of a blank space if the field doesn't contain an entry. The IIF( ) function takes the form

```
IIF(test expression, true result, false
    result)
```

If *test expression* evaluates to true, the function returns the *true result* argument; otherwise, it returns the *false result* parameter.

Testing your characters

Suppose you want to test the contents of a character field called NAME. To do so, you'd define a Formula Field and enter the expression

```
IIF(LEN(TRIM(NAME))>0, NAME,
    REPLICATE("_",LEN(NAME)))
```

When dBASE evaluates this expression, it returns the contents of the NAME field unless the field is blank. In that case, the expression LEN(NAME) tells the REPLICATE( ) function how many underline characters to return.

Zeroing in on missing values

So far, we've discussed testing only character fields with the IIF( ) function. If your database includes numeric fields, you'll need to adapt the arguments in the IIF( ) function accordingly.

For instance, the LEN( ) function returns only the length of character fields or expressions. So, you must find another way to specify the number of underlines for a numeric field.

You can use STR( ) to convert the numeric field, and then use LEN( ) to return the length of the resulting string. However, in the default dBASE environment, STR( ) returns a string of at least 10 characters, depending on the number of places you defined for the numeric field. If you're trying to keep your report columns as narrow as possible, 10 underlines may provide more room than your proofreader needs to update the report.

One approach is to hard-code in the Formula Field the number of underlines you want. For example, suppose you want to print five underscores for any zero values in a field named DWNPMT. To do so, you'd expect to enter a Formula Field using an expression in the form

```
IIF(DWNPMT<>0, DWNPMT, "____")
```

You can verify that this is a valid dBASE command by opening the SAMPLINE database and then entering

```
? IIF(DWNPMT<>0, DWNPMT, "____")
```

in the Command window. When you do, Visual dBASE evaluates the expression correctly. Specifically, dBASE will return either the numeric value in the DWNPMT field or a series of underlines. In this case, dBASE doesn't mind at all that you're mixing numeric and character data types as the IIF( ) function's second and third arguments.

Interestingly, Crystal Reports for dBASE lets you enter an IIF( ) function with mixed data types as the expression for a Formula Field. Then, when you click the Evaluate button, the program will display the correct result for the current record.

Unfortunately, at runtime, Crystal Reports for dBASE has a problem processing those mixed arguments in the IIF( ) function. When you attempt to print or preview the report, you'll receive the error message *Unknown error in dBASE expression.*

Fortunately, working around this problem requires just a little extra effort. When you enter the IIF( ) function, you simply use STR( ) to convert the numeric field to a string value. Of course, you can supply numeric arguments for the STR( ) function to determine how many characters it returns. However, for the sake of this example, we'll use the RIGHT( ) function to return the six rightmost characters in the string the STR( ) function returns.

Select Figure Field... from the Insert menu, name the field, and then enter

```
? IIF(DWNPMT<>0, RIGHT(STR(DWNPMT),6),"____")
```

If the DWNPMT field contains any value other than zero, this expression returns the six rightmost digits—including the decimal point, if any—from DWNPMT. If the DWNPMT field contains the value 0, this expression returns a series of underlines.

Dating your data

What happens if a data entry operator leaves an important date field blank? To identify a blank date field, you compare your field value to the expression { }. However, within the IIF( ) function, you must convert the date expression to character type, as you did for the DWNPMT field above. To create the Formula Field expression for a date field named PMTDATE, you'd enter

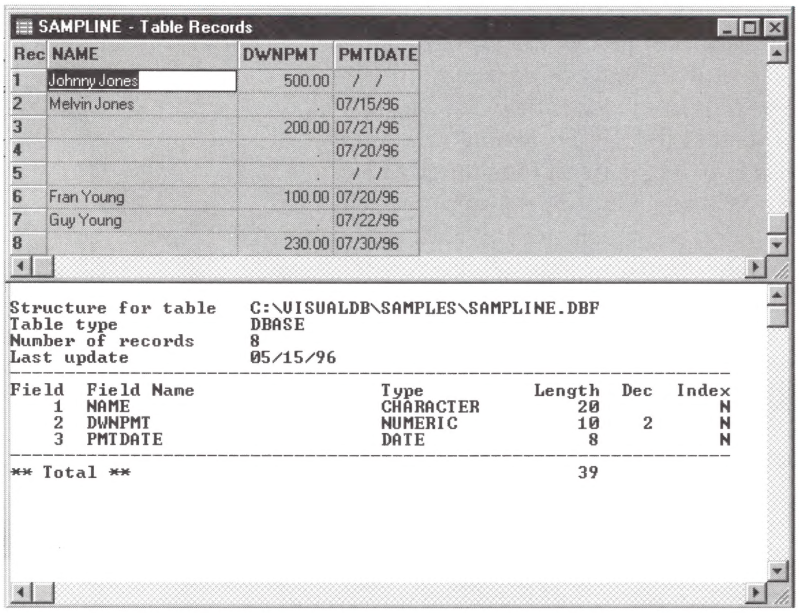
```
IIF(PMTDATE<>{ }, DTOC(PMTDATE), REPLICATE("_",8))
```

This expression returns eight underlines if the date field PMTDATE is blank. If the date isn't blank, the DTOC( ) function simply converts the date to a character string.

Creating the sample report

Now that you know what kinds of expressions you need, let's create a sample report. You can use any existing table, but to follow along with our example, you might want to create the SAMPLINE table with the records and structure shown in Figure C. Notice that many of the records contain fields with no entries, and several of the DWNPMT fields are blank.

Figure C

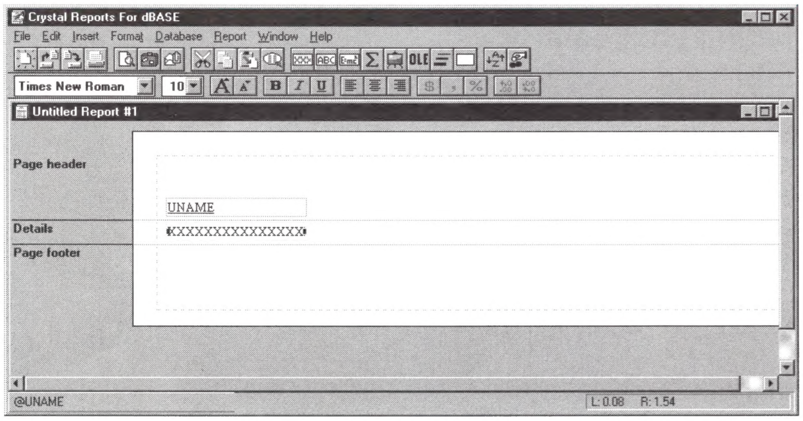


Rec	NAME	DWNPMT	PMTDATE
1	Johnny Jones	500.00	/ /
2	Melvin Jones	.	07/15/96
3		200.00	07/21/96
4		.	07/20/96
5		.	/ /
6	Fran Young	100.00	07/20/96
7	Guy Young	.	07/22/96
8		230.00	07/30/96

```
Structure for table    C:\VISUALDB\SAMPLES\SAMPLINE.DBF
Table type             DBASE
Number of records      8
Last update            05/15/96
----------------------------------------------------------------------
Field  Field Name                   Type          Length   Dec   Index
    1  NAME                         CHARACTER         20             N
    2  DWNPMT                       NUMERIC           10      2      N
    3  PMTDATE                      DATE               8             N
----------------------------------------------------------------------
** Total **                                           39
```

We'll use this table to create our sample report.

Figure D



This is how the sample report looks after you place the first Formula Field.

To create the report working from the Navigator, just select Reports, double-click the [Untitled] report icon, and then choose Designer. When the Insert Database Field dialog box appears, close it by clicking the Done button.

Next, open the Insert menu and choose the Formula Field… option. When the Insert Formula dialog box appears, type *UNAME* for the field name and click OK.

When the Expression Builder dialog box appears, enter

```
IIF(LEN(TRIM(NAME))>0, NAME,
    REPLICATE("_",LEN(NAME)))
```

Click the Evaluate button to make sure the expression contains no errors. Then click OK to close this dialog box. Next, use the mouse to drag the Formula Field's icon to the Details band. When you release the mouse button, the field mask and its label appear on the form, as shown in Figure D.

Now, you simply repeat the same process to place two more fields onto the report. Use *UPMT* to name the Formula Field for the DWNPMT field and enter the expression

```
IIF(DWNPMT<>0, RIGHT(STR(DWNPMT),6),"____")
```

Place the field in the Details band. Then, enter *UDATE* as the name for the Formula Field corresponding to the PMTDATE field. Enter the expression

```
IIF(PMTDATE<>{ }, DTOC(PMTDATE),
    REPLICATE("_",8))
```

and place this field on the Details band. When you run this report, it should look like the one shown in Figure B on page 8. (We added the title manually, since the Designer doesn't automatically add any text to the report's Page header.)

Conclusion

If your proofreader frequently overlooks missing fields when editing a dBASE report, you can redefine the fields in your report to print underlines instead of blank fields. In this article, we showed you how to use this technique to create reports that are easier to read. ❖

DATABASE TIP

Six guidelines for designing a database

by Susan Harkins

In last month's article "An Introduction to Data Normalization," we illustrated normalization issues by redesigning an inefficient database. However, redesigning a database you've already created is the worst way to normalize your data. The best database design begins before you ever launch Visual dBASE.

Designing a database is similar to writing a research paper—both projects will be easier and more functional if you create an outline first. That way, you can easily adjust your design on paper before problems make their way into your application. For the most part, designing a database is a six-step process that we'll outline for you in this article.

Step 1: Listing the data

To get started, list the pieces of data you plan to store in your database. For instance, let's suppose we want to create a database that stores retail product information. Initially, we can define four types of data: each product's name, price, and supplier, and each supplier's address.

After reviewing this data, it's easy to see that these four items fall into two categories: products and suppliers. So, early on, we've discovered that our database has two objectives, not just one—storing product information—as we originally thought. Now our objective is to store product *and* supplier information.

Step 2: Designing the tables

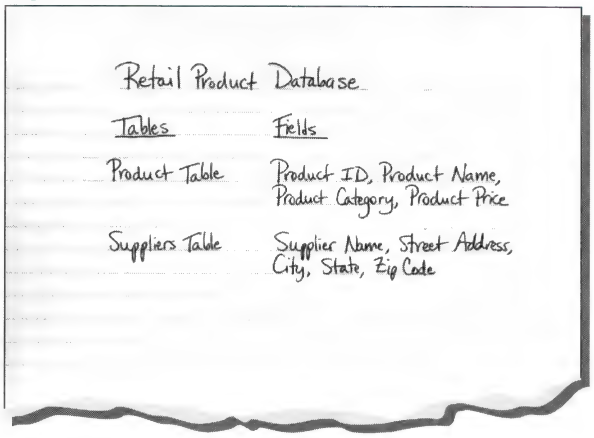
Now that we've ascertained that our database should have two tables—one for products and one for suppliers—we need to design each table. Listing all the data that belongs in each table will simplify this task.

As we mentioned, our product table should contain the name and price of each product—information that we can use later to group product types. In addition, we'll create a product category, which will include a product identification value to eliminate confusion if two products have the same name.

With these considerations in mind, we'll design the supplier table to contain the supplier's name, street address, city, state, and ZIP code. At this point, our list of required data might resemble that shown in Figure A.

Now, we have a good idea of the tables we'll need and what information we want to store in each one, so let's turn our paper lists into paper tables. To do so, simply list each field's type and length (where applicable), as we've done in Figure B.

Figure A

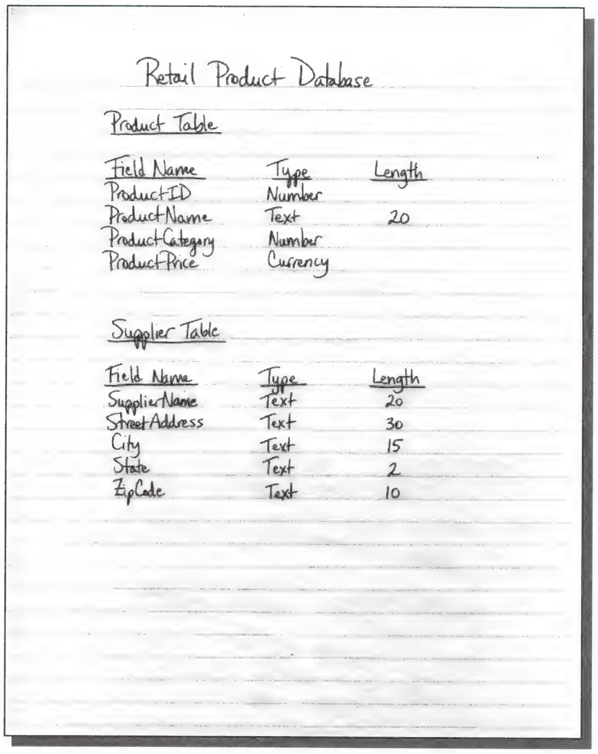


Retail Product Database

Tables	Fields
Product Table	Product ID, Product Name, Product Category, Product Price
Suppliers Table	Supplier Name, Street Address, City, State, Zip Code

First, we created a list of potential tables and their fields.

Figure B



Retail Product Database

Product Table

Field Name	Type	Length
ProductID	Number	
ProductName	Text	20
ProductCategory	Number	
ProductPrice	Currency	

Supplier Table

Field Name	Type	Length
SupplierName	Text	20
StreetAddress	Text	30
City	Text	15
State	Text	2
ZipCode	Text	10

Then, we listed the field information for each table.

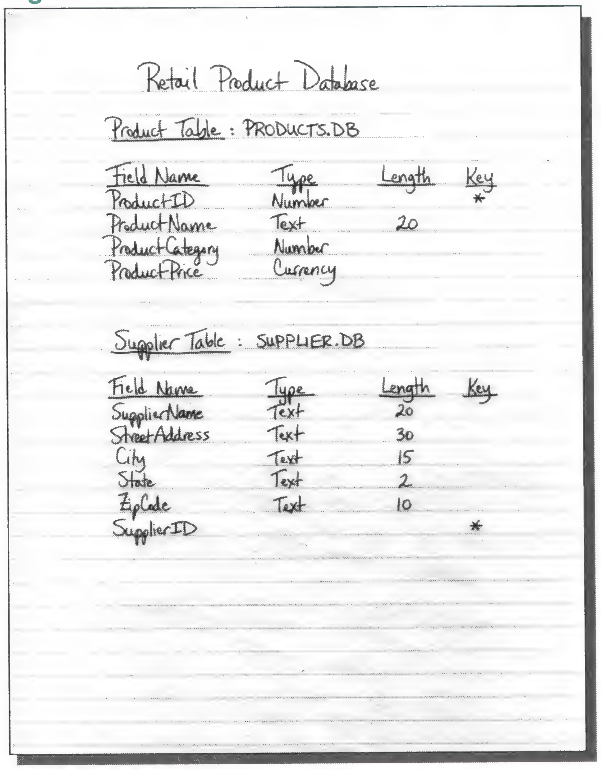
Step 3: Determining the relationships

Once you have a grasp of the data and the form it will take in your database, you can concentrate on how the data is related. The first thing we notice in our example is that each product has a supplier. Therefore, each product record in the product table (we'll name it PRODUCTS) should reference a supplier in the supplier table (SUPPLIER).

However, this isn't necessarily true in reverse. Each record in SUPPLIER won't always relate to a record in PRODUCTS. We can list many suppliers, even when we're not currently stocking any of their products.

In a relational database, we call this a *one-to-many relationship*. This means one table contains only one record for any number of records in another table. Each record in PRODUCTS refers to only one record in SUPPLIER; each record in SUPPLIER may refer to many records in PRODUCTS.

Figure C



Retail Product Database

Product Table : PRODUCTS.DB

Field Name	Type	Length	Key
ProductID	Number		*
ProductName	Text	20	
ProductCategory	Number		
ProductPrice	Currency		

Supplier Table : SUPPLIER.DB

Field Name	Type	Length	Key
SupplierName	Text	20	
StreetAddress	Text	30	
City	Text	15	
State	Text	2	
ZipCode	Text	10	
SupplierID			*

Next, we named our tables, added a supplier identification field to SUPPLIERS, and identified two primary key fields.

Step 4: Determining the keys

Once you've established a relationship between your tables, you must determine the field on which the relationship depends. This field is your primary key field. A *primary key* is a field that uniquely identifies a record. You'll note two distinctive characteristics:

- You can't store duplicate values in a primary key field.
- You can't have an empty primary key field.

Both our example tables have possible primary key candidates. The ProductID and ProductName fields in PRODUCTS should be unique. We can use either field as this table's primary key.

The SupplierName field in SUPPLIER is a good primary key candidate for that table, since the data is unique to each record. However, developers generally don't use a text field as a primary key field, because text fields are seldom unique. We can easily work around the text field, though, by adding a supplier identification field—similar to the ProductID field in PRODUCTS. We'll use this new field—SupplierID—as the table's primary key.

At this point, our table design has changed significantly, as shown in Figure C. We've added a new field and specified two primary key fields.

Step 5: Linking the tables

Once you know how your tables relate to one another, you need to link them. Doing so will protect the integrity of your data. Generally, you can't haphazardly delete and add records in related tables: You must follow certain rules. For instance, in our example we shouldn't delete a record from SUPPLIER if PRODUCTS contains a matching record.

It's at this point that we discover a flaw in our current design: There's no field common to both tables on which we can build a relationship. We can quickly repair this flaw, however, by adding the SUPPLIER table's SupplierID field to PRODUCTS. We can tell Visual dBASE to use PRODUCTS' SupplierID field to build a *secondary index*—an index based on a key other than the primary key—to sort and link the table. We can use this field to create a link between our two tables. We show our updated paper design in Figure D.

Foreign key

In our discussion of keys, we've mentioned only the primary key. Now that we've linked the two tables, we need to discuss a table's *foreign key*.

When you relate two tables, you have a parent table and a child table. The *parent table* contains the primary key, while the child table contains the *foreign key*—the primary key field in the parent table. In our database, SupplierID is the primary key in SUPPLIER and the foreign key in PRODUCTS.

If you want, you can enforce referential integrity when you relate tables. When you do so, you place the following constraints on the data:

- You can't add records to a child table if there's no associated record in the parent table.
- You can't delete records from the parent table when matching records exist in the child table.

In our example database, this means

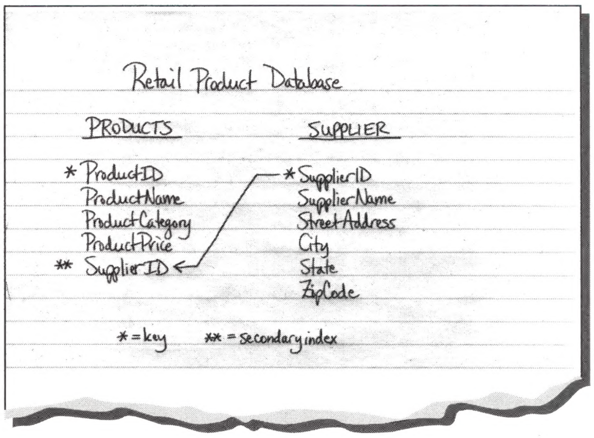
- We can't add a record to PRODUCTS if the specified SupplierID value doesn't exist in SUPPLIER.
- We can't delete a supplier's record in SUPPLIER if any record in PRODUCTS contains that supplier's SupplierID value.

Step 6: Evaluating the design

At this point, we've completed a rough draft of our database. Now is the time—*before* we implement the design—to evaluate our work. Fortunately, you can follow a few simple guidelines to help you assess your database design. Be sure that:

- Each field in a table relates to its key field.
- Each table has a key—primary or foreign.

Figure D



Finally, we linked the tables on the SupplierID field.

- Each table is restricted in a logical manner.

In our example, the fields in each table relate to that particular table's key. SUPPLIER's primary key is the SupplierID field—this field relates to each field of supplier data in this table.

PRODUCTS' primary key is ProductID, which relates to each field of product data. In addition, PRODUCTS contains the foreign key, SupplierID. This key doesn't relate as strongly to the product fields, but it does relate.

We've restricted both tables by enforcing referential integrity on the contents of the SupplierID field. This is the only logical restriction we can make, since the SupplierID field is present in both tables.

Putting the design to work

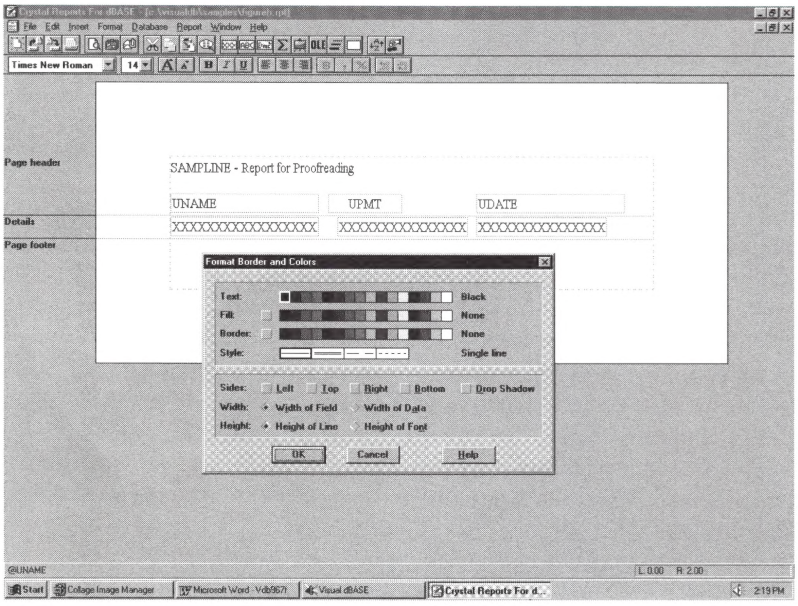
We've designed a simple database to illustrate six useful design guidelines. In a larger database with many tables, you'll find your work a little more complicated; however, the same rules apply. You'll find the finished product much easier to use and more dependable if you design your database before you build it. ❖

Visual dBASE Specific

Enhancing reports by adding field borders

You can let your imagination run wild when you design a report with Crystal Reports for dBASE. In this article, we'll show you how to enhance a report by drawing borders around some or all of its data items.

Figure A



Crystal Reports For dBASE - [c:\visualdb\samples\figureb.rpt]

Page header: SAMPLINE - Report for Proofreading

UNAME UPMT UDATE

Details: XXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX

Page footer

Format Border and Colors

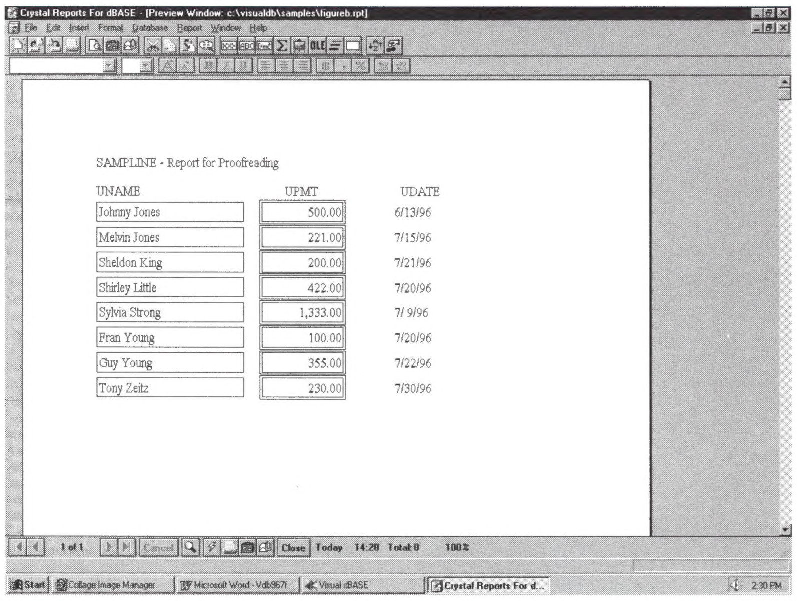
Text: Black
Fill: None
Border: None
Style: Single line

Sides: Left, Top, Right, Bottom, Drop Shadow
Width: [✓ Width of Field] Width of Data
Height: [✓ Height of Line] Height of Font

File Edit Insert Format Database Report Window Help
Times New Roman 14
OK Cancel Help
@UNAME
L 0.00 R 2.00
Start Collage Image Manager Microsoft Word - Vdb967l Visual dBASE Crystal Reports For d...
2:19 PM

You'll use this dialog box to add borders, among other formatting options, to the fields in your report.

Figure B



Crystal Reports For dBASE - [Preview Window: c:\visualdb\samples\figureb.rpt]

SAMPLINE - Report for Proofreading

UNAME	UPMT	UDATE
Johnny Jones	500.00	6/13/96
Melvin Jones	221.00	7/15/96
Sheldon King	200.00	7/21/96
Shirley Little	422.00	7/20/96
Sylvia Strong	1,333.00	7/ 9/96
Fran Young	100.00	7/20/96
Guy Young	355.00	7/22/96
Tony Zeitz	230.00	7/30/96

File Edit Insert Format Database Report Window Help
1 of 1 Cancel Close Today 14:28 Total: 0 100%
Start Collage Image Manager Microsoft Word - Vdb967l Visual dBASE Crystal Reports For d...
2:30 PM

Borders create eye-catching effects for your reports.

On the borderline

You can demonstrate our technique by right-clicking on any report form and choosing the Design Report option. When the report layout appears, click on a field mask to select it. You can open the Format menu and choose the Border And Colors... option or right-click on the field mask and choose that option. When you do, the Format Border And Colors dialog box appears, as shown in **Figure A**.

Simply click on one of the icons next to the *Style* label. You can choose from among single, double, dashed, and dotted-line styles. In addition, you can specify whether you want the border to appear around the entire field or around the data itself. Select the single-line style option and click OK.

To preview your report, select Print from the File menu and choose the Window option. **Figure B** shows our sample report after we added a single-line border to the UNAME field in the first column and a double-line border to the UPMT field in the second column.

Conclusion

In this article, we've shown you how to add borders to the fields in your reports by using Crystal Reports for dBASE's formatting options. If you use color printers, or if you develop applications that preview reports to the screen exclusively, you can expand on this technique to enhance your reports with custom color schemes for your different fields. ❖

Send us your favorite reports

Have you created a particularly impressive report using Crystal Reports for dBASE? If so, we want to hear from you! Write us and describe your technique in detail, or send us a copy of your report and its source database. If we publish an article featuring your report technique, we'll send you a check for $25 to show our appreciation. (See the masthead on page 15 for our address.)

Modifying the effect of pressing [Enter]

In dBASE for DOS applications, you use @...SAY...GET commands to display a message and an input field for data entry. Then, the operator typically moves the cursor between fields in the entry screen by pressing [Enter]. Of course, you can also move the cursor between fields in a data entry screen by typing an entry that fills the entire field or by pressing a cursor-movement key.

If your users are accustomed to Windows-style applications, they'll probably have no trouble getting accustomed to pressing [Tab] to move between fields and pressing [Enter] to close a form or to select a form's default pushbutton. However, suppose you're migrating a DOS application to the Windows environment, but you want to maintain consistency with the way the DOS application handles the [Enter] key. In this article, we'll show you a command that lets you decide whether the [Enter] key closes a form or terminates data entry in the current field and moves the cursor to the next screen object in the form.

The SET CUAENTER command

The key to this technique is the SET CUAENTER command, which you can set ON or OFF. The default setting for SET CUAENTER is ON. To change the default, you can update the CUAENTER setting in dBASEWIN.INI. To do so, either use the SET command to specify the setting interactively, or edit the CUAENTER parameter directly in dBASEWIN.INI. To tell Visual dBASE to treat the [Enter] key in the dBASE for DOS manner, simply enter the command *SET CUAENTER OFF.* ❖

INSIDE VISUAL dBASE™

Inside Visual dBASE (ISSN 1084-1970) is published monthly by The Cobb Group.

Staff:

Contributing Editor-in-Chief	Keith G. Chuvala
Associate Editors-in-Chief	Jeff E. Davis
	Tiffany M. Taylor
Publications Coordinator	Maureen Spencer
Editors	Laura Merrill
	Joan McKim
	Elisabeth Pehlke
Production Artist	Alison Schwarz
Product Group Manager	Mike Stephens
Circulation Manager	Mike Schroeder
Associate Publisher	Mark Kimbell
VP/Publisher	Mark Crane
President/CEO	J. Thomas Cottingham

Address: Please send tips, special requests, and other correspondence to

The Editor, *Inside Visual dBASE*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

E-mail address: visual_dbase_win@merlin.cobb.zd.com

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

E-mail address: customer_relations@merlin.cobb.zd.com

Phone:

Toll free US	(800) 223-8720
Toll free UK	(0800) 961897
Local	(502) 493-3300
Customer Relations Fax	(502) 491-8050
Editorial Department Fax	(502) 491-3433

Back Issues: To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $8.50 each, $8.95 outside the US. You can pay with MasterCard, VISA, Discover, or American Express, or we can bill you.

Copyright: Copyright © 1996, The Cobb Group. All rights reserved. *Inside Visual dBASE* is an independently produced publication of The Cobb Group. The Cobb Group reserves the right, with respect to submissions, to revise, republish, and authorize its readers to use the tips submitted for personal and commercial use.

The Cobb Group and its logo are registered trademarks of Ziff-Davis Publishing Company. *Inside Visual dBASE* is a trademark of Ziff-Davis Publishing Company. dBASE, dBASE IV, dBASE for Windows, and Visual dBASE are registered trademarks of Borland International. Windows is a registered trademark of Microsoft.

Postmaster: Second Class Postage Paid in Louisville, KY.

Postmaster: Send address changes to:

Inside Visual dBASE
P.O. Box 35160
Louisville, KY 40232

Advertising: For information about advertising in Cobb Group journals, contact Tracee Bell Troutt at (800) 223-8720, ext. 430.

Bulk Sales: For information about bulk/group subscription sales, please contact Customer Relations at (800) 223-8720.

Prices:

Domestic	$79/yr ($8.50 each)
Outside US	$89/yr ($8.95 each)

SECOND CLASS MAIL

Please include account number from label with any correspondence.

Help shape the future of *Inside Visual dBASE*

The Cobb Group wants *Inside Visual dBASE* to meet your needs. Your input is vitally important to the success of this journal, so we're asking you to tell us how we're doing. Please help us get to know you better by taking a few moments to fill out this survey. Just photocopy this page, fill it out, and mail or fax it to us by September 1, 1996. We'll hold a random drawing from all the responses, and three winners will receive a free one-year subscription to *Inside Visual dBASE*.

Return surveys to:
Inside Visual dBASE
The Cobb Group
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Or fax this page to:
Inside Visual dBASE
(502) 491-3433

You can also contact *Inside Visual dBASE* via E-mail, and we'll put your name in the drawing for the one-year subscriptions. Just send your comments to

`visual_dbase_win@merlin.cobb.zd.com`

CompuServe members can address comments to 76260,1766. Thanks for your input! We'll publish the survey results in a future issue.

How do *you* use Visual dBASE?

Your job title ______________________

How do you rate your experience with Visual dBASE?
❏ Novice ❏ Advanced ❏ Expert

Please describe how you use Visual dBASE:
❏ I use programs written by someone else.
❏ I design databases and reports but do no programming.
❏ I write programs for my own use.
❏ I develop applications for use by others.

Which version of dBASE do you use?
❏ Exclusively dBASE for Windows 5.0
❏ Some dBASE for Windows 5.0, some Visual dBASE 5.5
❏ Exclusively Visual dBASE 5.5

If you use dBASE for Windows 5.0, do you plan to upgrade to Visual dBASE?
❏ Yes ❏ No If Yes, when? ______________

How would you rate the technical level of the articles in *Inside Visual dBASE*?
❏ Most are above my skill level
❏ Most are at my skill level
❏ Most are below my skill level

All things considered, how satisfied are you with your subscription to *Inside Visual dBASE*?
❏ Very satisfied ❏ Dissatisfied
❏ Satisfied ❏ Undecided

What article topics would you like to see in *Inside Visual dBASE*? (Please be specific.)

____
____
____
____
____
____
____

What other computer publications do you read?

____
____
____

Which, if any, of these online services do you use?
❏ America Online ❏ Internet access
❏ CompuServe ❏ MSN
❏ Genie ❏ Other ______________

7/96

Printed in USA
This journal is printed on recyclable paper.